Technical Disclosure Commons

Defensive Publications Series

February 03, 2016

METHOD FOR ENSURING DATA CORRECTNESS BETWEEN CLIENT AND SERVER WITH A LIMITED CLIENT IMPLEMENTATION

Sorin Mocanu

METHOD FOR ENSURING DATA CORRECTNESS BETWEEN CLIENT AND SERVER WITH A LIMITED CLIENT IMPLEMENTATION

ABSTRACT

A method for ensuring data correctness between client and server with a limited client implementation in a calendar application is disclosed. The client runs an application, which supports creating and editing a limited number of simple scenarios. Through interoperability with third party application programming interface (APIs) events, further complex recurring rules can be created. In one instance, the application is implemented with a trimmed down recurrence library for allowing expansion of canonical events to the list of instances directly by the client. The application for ensuring data correctness between client and server using the method illustrated can be implemented in any calendaring applications.

BACKGROUND

Mobile-first rich clients need to include domain or business logic in order to ensure a correct view of locally mutated data even when network connectivity is poor or unavailable. One of the problems with relevance to calendar application is allowing users to edit recurring events offline. Recurring events are events that happen repeatedly based on a metarule defined by the scheme for calendar interoperability (RFC 2445 or RFC 5445). A calendar application backend may strive for support beyond the rules supported by RFC 2445 with some theoretically impossible combinations (e.g. WEEKLY recurrence with BYWEEKDAY clauses). Existing libraries, such as ical4j, support these expansions. Such libraries may be deemed to be too

complex and heavyweight for running in the performance-critical sections of a client application such as a mobile app.

In an online calendar application, fully correct data is rendered by doing the relevant calculations (recurrence expansions) on the server side. Whenever an event is modified, the changes are sent to the server for calculating values to be rendered in the client application. Further, server rendering of this data requires connectivity and sufficient data traffic before the change can be reflected on the device.

In a mobile or offline-capable application, user has to rely on the application's own built-in recurrence rule library to see the times of a particular recurrence rule without communicating with the server. The client library may produce expansions that may be inconsistent with the backend, leading to differences between instances of recurring events reported by the backend and the device. The user may therefore have to compromise on data correctness. Supporting all these scenarios in a client application may not be feasible. Thus there is a need of a calendar software application that can reflect user modified recurring events even when the user is on a flaky network or is disconnected.

DESCRIPTION

The goal is to create a method for ensuring data correctness between client and server with a limited and intermittently performant client implementation in a software application. The client will run an application, which supports creating and editing a limited number of simple

scenarios. Through interoperability with third party application programming interface (APIs) events, further complex recurring rules can be created.

In one instance, the application will be implemented with a trimmed down recurrence library for allowing expansion of canonical events to the list of instances directly by the client. In this recurrence library, to keep the process of implementation simple, only essential features of the scheme for calendar interoperability (RFC 2445 or RFC 5445) are included.

The interpretation of the recurrence rules are executed as follows:

- Creating and editing rules by the client, which can be expanded on the client
- Sharing a library by the server and client with defined rules, which can be expanded on the client
- Converting between a RFC RRULE string and a structured format by the server for easy parsing by the client
- Sending simple recurring rules in a canonical structured format by the server to the client when the server confirms that the client can recognize them
- Expanding the exact dates for complex recurring rules by the server and sending denormalized data to the client
- Rendering events at the appropriate points in time based on,
    a. structured format for simple recurring rules or
    b. exact dates sent by the server for complex recurring rules

Therefore, the application supports a broad range of features (in the calendar case, understanding any recurrence rule), without client implementation of long tail or highly

specialized library and unlikely corner cases. The application for ensuring data correctness between client and server and the system using the method illustrated can be implemented in any calendaring application.

The advantages of using the method include ensuring data correctness between client and server with a limited client implementation in the client application. Further, this method can develop and maintain a new user-friendly business logic library because the more complex cases will resort to server resolution.